PC 7
Graphes

1. représentations
2. matrice d'adjacence
3. fermeture transitive
4. exploration
5. plus court chemin

Introduction

Un graphe (S, F) est donné par

- Un ensemble S de sommets
- Un ensemble $F \subset S \times S$ de *flèches* (ou arcs).

Un *chemin* dans G est une suite

$$(s_0, s_1), (s_1, s_2), \ldots, (s_{n-1}, s_n)$$

Variantes:

- On a aussi en général des *étiquettes* sur les flèches ou sur les sommets.
- Graphes non orientés.
- Graphes représentés (dans le plan par ex.)

Exemples de graphes

Les graphes se rencontrent comme:

1. Modèle de liaisons spatiales: réseaux de transport, électriques, ...
2. Représentations de liens logiques: graphes de dépendance.
3. Modèles de l'évolution temporelle: les sommets sont des *états* d'un système.
4. Schémas de programmes: les sommets sont les instructions.

Exemples

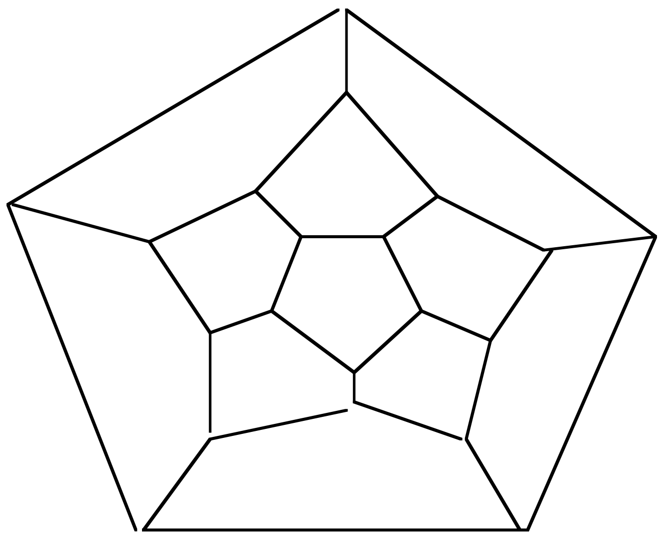

Figure 1: dodecaèdre

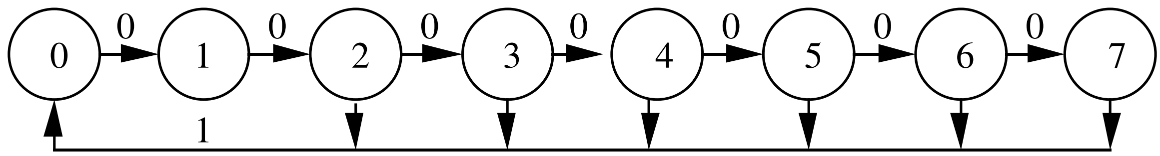

Figure 2: canal contraint

Représentation des graphes

1. Matrice d'adjacence:

$$A[i,j] = \begin{cases} 1 & \text{si } (i,j) \in F \\ 0 & \text{sinon} \end{cases}$$

   Les éléments peuvent être considérés comme des booléens ou comme des entiers.

2. Liste de successeurs: On donne pour chaque sommet la liste de ses successeurs:

$$L(s) = \{t \in S \mid (s,t) \in S\}$$

3. Matrice d'incidence: On donne pour chaque flèche son origine et son extremité:

$$I[i,f] = \begin{cases} 1 & \text{si } f = (i,.) \\ -1 & \text{si } f = (.,i) \\ 0 & \text{sinon} \end{cases}$$

Exemple

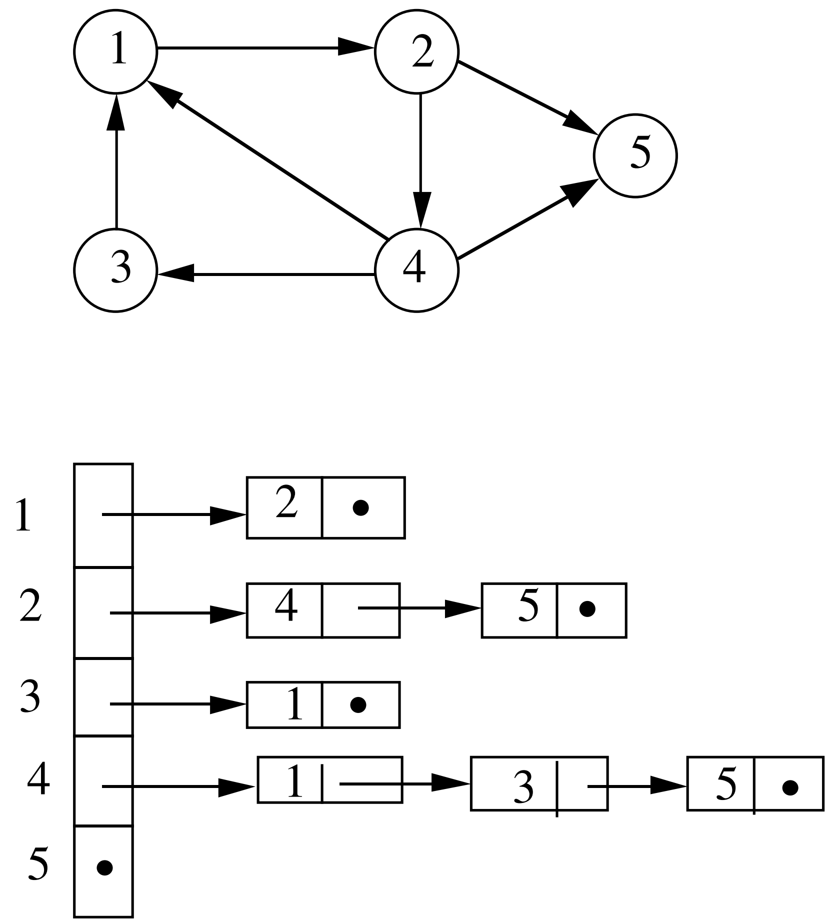

Figure 3:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Fermeture transitive

On se donne un graphe $G = (S, F)$ par sa matrice d'adjacence A.

On veut calculer la *fermeture transitive* de G, i.e. le graphe $G^* = (S, F^*)$ défini par $(s, t) \in F^*$ ssi

il existe un chemin de s à t dans G

Propriété fondamentale de la matrice d'adjacence:

$$\boxed{A^k[i,j]=\text{nb de chemins de longueur } k \text{ de } i \text{ à } j}$$

ou en booléens

$$\boxed{A^k[i,j] = 1 \Leftrightarrow \exists \ \begin{array}{r}\text{chemin de longueur } k \\ \text{de } i \text{ à } j\end{array}}$$

Formule donnant la matrice d'adjacence A^* de G^*(en booléens):

$$\begin{aligned} A^* &= I + A + A^2 + \ldots + A^{n-1} \\ &= (I + A)^{n-1} \end{aligned}$$

puisqu'un chemin de i à j peut être choisi de longueur au plus $n - 1$.

Algorithme naïf

L'algorithme brutal pour calculer la matrice d'adjacence T de G^* est en $O(n^4)$:

On initialise B et T à I. Si `produit(A,B)` remplace B par AB et `somme(B,T)` remplace T par $B+T$ on écrit

```
procedure Fermeture(A: matrice;
                var T:matrice);
 var i,j,k: integer;
     B: matrice;
 begin
  identite(B); identite(T);
  for k:= 1 to n-1 do begin
   produit(A,B);
   somme(B,T);
  end;
 end;
```

L'algorithme de Warshall

Idée: au lieu de faire porter la récurrence sur la longueur du chemin, on peut la faire porter sur le nombre de sommets utilisés. Posons:

$$A^{(k)}[i,j] = \begin{cases} 1 & \text{s'il existe un chemin de i à j} \\ & \text{ne passant par} \\ & \text{aucun sommet} > k \\ 0 & \text{sinon} \end{cases}$$

L'algorithme est basé sur la formule de récurrence (en booléens):

$$A^{(k)}[i,j] = A^{(k-1)}[i,j] + A^{(k-1)}[i,k]A^{(k-1)}[k,j]$$

Programme en pascal:

```
for k:= 1 to n do
 for i:= 1 to n do
  for j:= 1 to n do
   T[i,j]:=T[i,j]
           or
           (T[i,k] and T[k,j]);
```

Plus court chemin

On se donne sur chaque flèche(i, j) une *longueur* $l(i, j)$ positive. On cherche à déterminer le plus court chemin d'un sommet à un autre.

On représente le graphe par une matrice C définie par

$$C[i, j] = \begin{cases} l(i, j) & \text{si } (i, j) \in F \\ 0 & \text{si } i = j \\ \infty & sinon \end{cases}$$

On pose ensuite $C^{(k)}[i, j] =$ la longueur du plus court chemin de i à j utilisant au plus k flèches.Du fait que les longueurs sont positives, on a la relation de récurrence:

$$\boxed{C^{(k+1)}[i, j] = \min \begin{cases} C^{(k)}[i, j] \\ \min \begin{array}{l} \{C^{(k)}[i, u] + C[u, j] \\ \mid 1 \leq u \leq n\} \end{array} \end{cases}}$$

Ceci conduit à un algorithme en $O(n^4)$.

L'algorithme de Floyd

On considère cette fois $C_k[i,j]$ = la longueur du plus court chemin de i à j ne passant par aucun sommet $> k$. On a la relation de récurrence:

$$\boxed{C_{k+1}[i,j] = \min \begin{cases} C_k[i,j] \\ C_k[i,k+1] + C_k[k+1,j] \end{cases}}$$

d'où le programme:

```
for k:= 1 to n do
 for i:= 1 to n do
  for j:= 1 to n do
   if A[i,k]+A[k,j]<A[i,j] then
    begin
      A[i,j]:= A[i,k]+A[k,j];
      P[i,j]:= k
    end;
```

Le tableau P sert à retrouver le plus court chemin. Il est initialisé à 0.

```
procedure chemin (i,j: integer);
 var
  k: integer;
 begin
  k:= P[i,j];
  if k <> 0 then
    begin
     chemin(i,k);
     writeln(k);
     chemin(k,j)
    end
  end;
```

Semianneaux

On peut faire les mêmes calculs dans des structures algébriques différentes.

ensemble	addition	mult	zero	unité
booléens	ou	et	faux	vrai
entiers naturels	$+$	$\times$	0	1
réels pos.	min	$+$	∞	0

Un semianneau est la même chose qu'un anneau mais sans $-$:

1. semigroupe pour l'addition avec neutre 0
2. semigoupe pour la multiplication avec neutre 1
3. distributivité de $+$ sur $\times$.